Automated image processing tools for high throughput measurements of polymer coatings: initial report on software to quantify features

Isabel Beichl
Jane Bernstein
and
Alamgir Karim

1 Introduction

We have developed a series of Matlab programs to analyze photographs of polymer dewetting. This report provides details of how the programs work, what they do, and how users can fine tune the output.

In [1], Meredith et al. have described a method to gather massive amounts of data on the dewetting process for polymers, by using a combinatorial approach to data collection. These data, produced by automated microscopy, are collected in the form of photographs of polymers varying the parameters of temperature, thickness and time. The purpose of the tools described in this report is to identify features of the dewetting process and to quantify the features *without* having a human doing the feature recognition or the counting. The volume of the data makes human processing utterly impractical. In this report we describe methods used on photographs of three stages of dewetting. This report is preliminary. A set of tools to automate the analysis of the photographic images completely, without regard to the state of dewetting, is the ultimate goal.

2 Feature Recognition and Image Cleaning

Some examples of dewetting photographs examined in this report are shown in figures 1, 2, 3 and 4.

For each of these types of images, we give methods for cleaning, extracting and quantifying features. For each case we also compute a voronoi diagram, to be explained later, and suggest it as a possible description of the evolution of one state to the next.

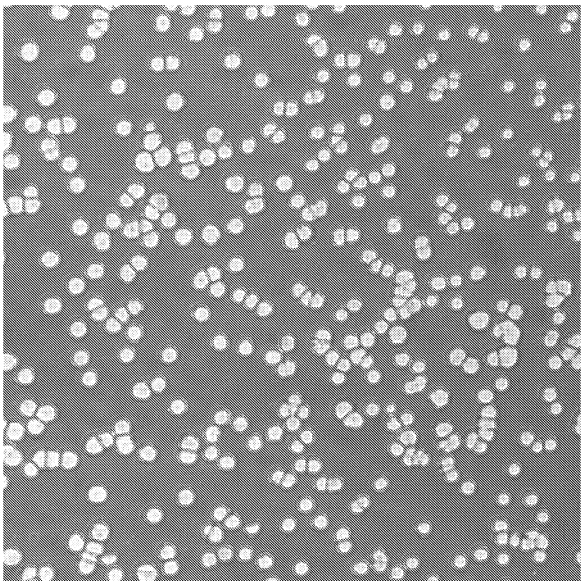

Figure 1: A stage of dewetting characterized by circular holes.

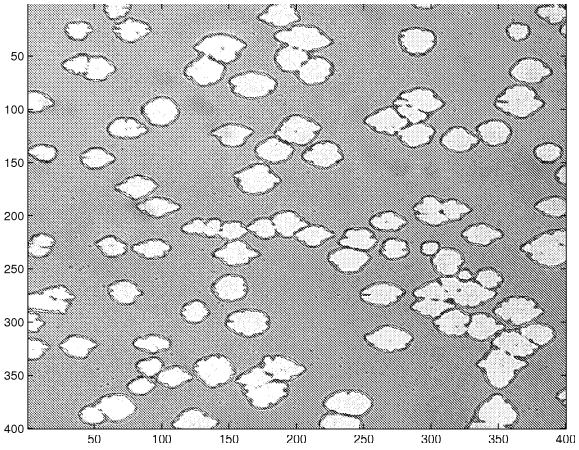

Figure 2: A stage of dewetting characterized by asymmetric holes.

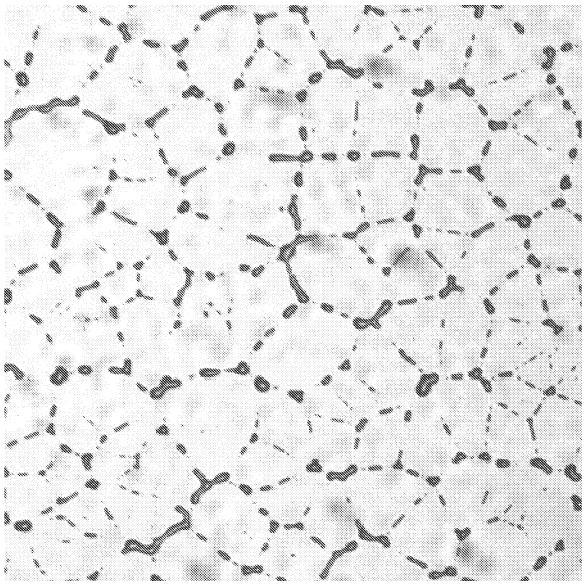

Figure 3: A stage of dewetting characterized by material forming beads that resemble polygons.

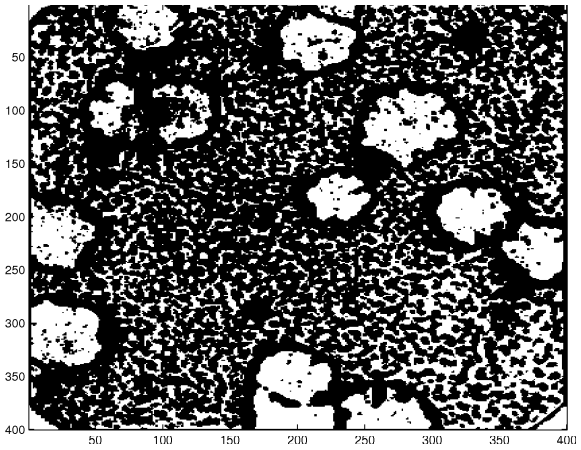

Figure 4: Not only does hole size vary but the extent of cleaning needed to extract data varies also.

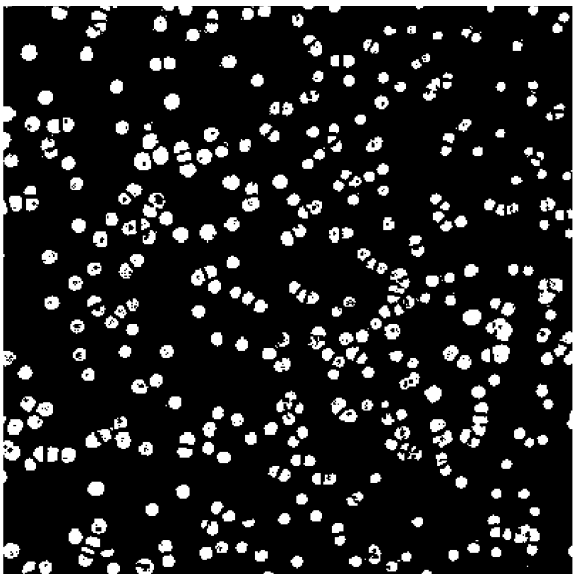

Figure 5: Image from figure 1 after thresholding.

3 Thresholding and Cleaning

The photographs are in a digitized "tif" format which matlab can handle. This allows them to be input as a matrix of integers between 0 and 256 representing a degree of grayness from black to white. Because the images come from real experiments there are imperfections that need to be removed before one can quantify the features because the imperfections will be interpreted as features. It is easier to remove these unwanted effects if we first threshold the image, that is to define a range of brightness that is considered to be the substrate. In this way the image is transformed into a binary image, that is a matrix of 0's and 1's. However because there is a temperature gradient across the surface of the experiment, the range of brightness considered to be substrate in one part of the image is not the same everywhere. We approximate the threshold by a linear function that is known at the corners of the image. Figure 5 illustrates the result of this operation on the "Circular Hole" data set from Figure 1. (Another technique which we have also used but not included in the code is using the matlab "gradient" function on the data before thresholding and in this way replace each pixel with the difference between it and its four neighbors. This also removes some of the effects of the temperature gradient on the data and allows a constant threshold.)

After thresholding, it is necessary to "clean" the image to remove stray bits that we do not consider part of the features. These occur both on the substrate and on the material. We use a combination of the "morphing" capabilities of matlab to accomplish this. Figure 6 illustrates the result of these operations. It

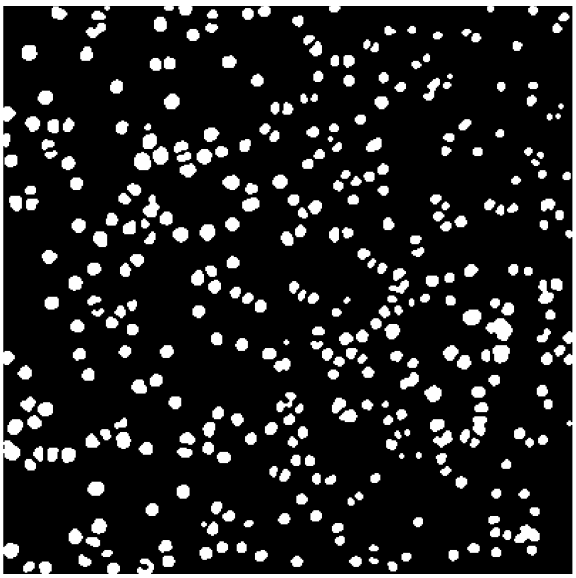

Figure 6: Image from figure 5 after cleaning.

may be necessary, depending on the data, to repeat these "morphing" operations because each invocation of these operations makes the boundary more uniform, more smooth. The number of re-invocations of the morphing commands to obtain optimal solutions is not known although for the "ideal" data given in figures 1-3, one invocation is sufficient. The code in the appendix uses the matlab comands, *bwmorph* and $bwfill$, for cleaning.

4 Geometric Characteristics of "Holes"

After thresholding and cleaning the image, the code uses *bwlabel* to identify the connected components of the white space. It does this by assigning different identifying numbers to different areas that are unconnected. We use these numbers to obtain information about the holes themselves, that is the white area of Figure 6. So in the code we use *bwlabel* to obtain identifying numbers and $max(max(dd))$ to count them. Similarly, we can count the number of pixels assigned to each "hole" and by dividing by the total number of pixels in the image we can get estimates of the area of each hole relative to the size of the image. After we have done this, making histograms is possible. This is illustrated in figure 7. For the case of dewetting states such as figure 3, the areas of interest are the polymer itself and not the substrate. Thus to use these techniques on this data, we first need to exchange black and white which is done in matlab with the command $a = \tilde{a}$ for image a.

Similar to area, the perimeters of the holes can be computed and histograms

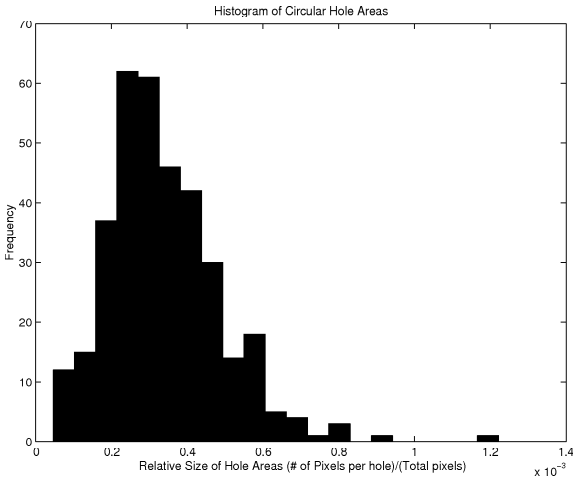

Figure 7: Histograms of relative area of the "holes" from figure 6.

can be made with the *bwperim* function which detects the perimeter. Figure 8 illustrates the perimeters of the data from figure 6.

We would like to characterize the extent to which a hole deviates from being a perfect circle without reference to its size. This is possible by computing the *circularity* of the hole which by definition is the square of the perimeter divided by the area. For a perfect circle of radius, R, the circularity is always $Perim^2/Area = (2*\pi*R)^2/(\pi*R^2) = 4*\pi$. Any other hole will have a greater circularity. Because the holes in figure 6 are very circular they do not make an interesting test case. However the data from figure 2 is not so regular. A histogram of the circularity of the asymmetric holes is shown in figure 9.

We also compute centers and radii for the holes computed by *bwlabel*. We do this by finding the row in the image where a given hole covers the maximum number of columns. This assumes that the holes, if eccentric, have axes parallel to the boundaries of the image. This assumption is reasonable because the temperatures and thickness gradients are always parallel to the boundaries of the plate. We are able to compute histograms for these cases too. Once we have an estimate for the centers of the cells we are able to look at the distribution of the centers throughout an image. This is shown in the code in the appendix in the section on "HOLE DENSITY". We divide the image into a certain number of bins and count the number of hole centers that appear in a bin. As expected, the distribution of these holes is a Poisson distribution. We compute the mean in the section of the code labeled "Poisson Distribution".

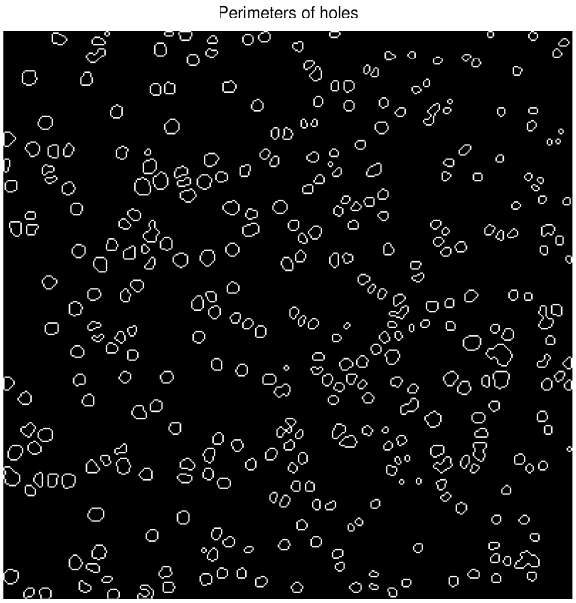

Figure 8: Perimeters of holes in figure 6.

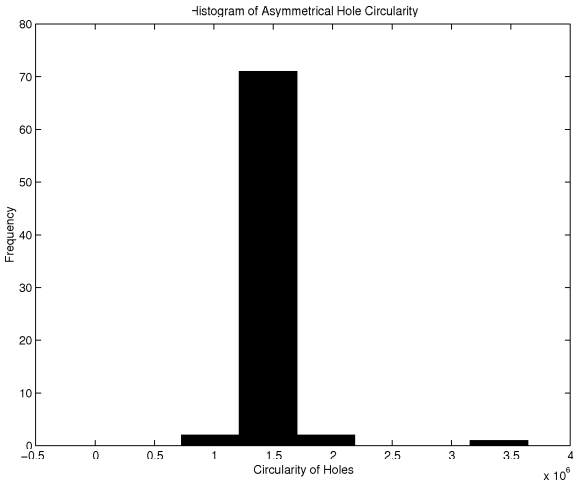

Figure 9: Histograms of circularity derived from data in figure 2.

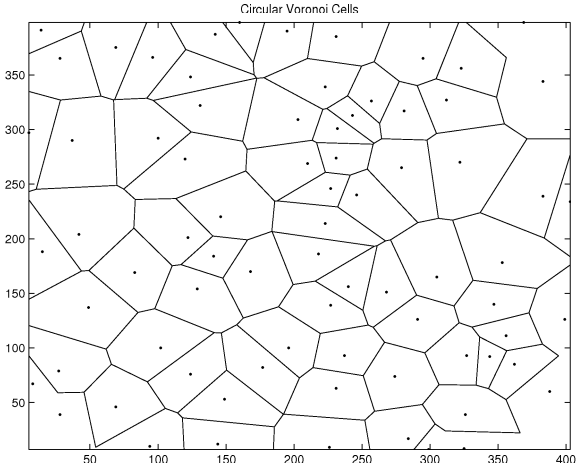

Figure 10: Voronoi Diagram computed from the centers of holes from figure 6.

5 Evolution

Lastly, we compute the Voronoi diagram of the centers obtained. For each center, now considered as a vertex, a polygon is computed. The polygons do not intersect except at their boundaries. The Voronoi diagram is a set of polygons that divide up the plane so that a given polygon is the set of points in the plane closest to the vertex associated with that polygon. An example is shown in figure 10. We compute the size of voronoi cells and make histograms. We are also able to compute the number of edges in a given voronoi cell and the average length of an edge. The code for each of these functions can be found in the appendix under "VORONOI CELLS". We suspect that as dewetting proceeds, the material tends to evolve into polygons in the shape of the Voronoi diagram of its original hole centers. This is a topic for further investigation as to the extent that dewetting actually proceeds in this way.

Disclaimer

Certain commercial software may be identified in order to adequately specify or describe the subject matter of this work. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the equipment identified is necessarily the best available for the purpose.

References

[1] J.C.Meredith, A.P.Smith, A.Karim, E.J.Amis, "Combinatorial Material Science for Polymer Thin Film Dewetting", "Combinatorial Materials Science for Polymer Thin-Film Dewetting", *Macromolecules*, vol. 33, 9747-9756 (2000).

[2] J.C.Russ *The Image Processing Handbook* CRC Press, (1992).

6 Appendix: Matlab Code for Extracting Features for Circular Hole State

```
%this matlab program produces an image as a 400x400 matrix.
%It reads in from a ``tif'' file and makes a histogram of the pixel values.
%Then it manipulates the matrix, and thresholds the image.  Then, it
%cleans the image, prints it to the screen and calculates the number of holes
%in the image.  It creates a histogram of the hole areas, perimeter, diameters,
%and circularities.  It also finds the centers of the holes. Finally, it
%calculates hole densities in the image and makes histograms of the hole
%distributions.

x=imread('filename','tif');
%input is taken from a tif file named 'filename'
xx=double(x);

xa=min(min(xx)):(max(max(xx))-min(min(xx)))/15:max(max(xx));
hist(xx,xa);
title('Histogram of Original Pixel Values');
xlabel('Pixel Values of Matrix');
ylabel('Frequency');

m=min(min(xx));
xx=xx-m;
m=(105-140)/(400-1);
for i=1:400
  for j=1:400
    thresh(i,j)=m*(j-400)+110; %thresholding image
  end
end
for i=1:400
  for j=1:400
    if xx(i,j)>thresh(i,j)
yy(i,j)=1;
    else
yy(i,j)=0;
```

```
    end;
  end
end
aa=bwmorph(yy,'clean',Inf);
bb=bwmorph(aa,'majority',Inf); %cleaning up the image
cc=bwmorph(bb,'fill',Inf);
zz=bwfill(cc,'holes');
dd=bwlabel(zz);

imshow(dd);
title('Circular Holes');

num_holes=max(max(dd))

total_area=0;
total_perim=0;
[r,c]=size(xx);

for i=1:max(max(dd))
hole_area=(sum(sum(dd==i))); %area of each hole
area_array(i)=hole_area/(r*c);
total_area=total_area+hole_area;
end

total_area
fracwhole_area=total_area/(r*c)

x=min(area_array(1:max(max(dd)))):(max(area_array(1:max(max(dd))))-
    min(area_array(1:max(max(dd))))/20:max(area_array(1:max(max(dd))));
%20 is number of bins
hist(area_array(1:max(max(dd))),x);
title('Histogram of Circular Hole Areas'); %area histogram
xlabel('Size of Hole Areas (# of Pixels)');
ylabel('Frequency');

%PERIMETER

ee=bwperim(dd);
ff=bwlabel(ee);
for i=1:max(max(ff))
hole_perim=(sum(sum(ff==i))); %perimeter of each hole
perim_array(i)=hole_perim;
total_perim=total_perim+hole_perim;
end

total_perim
```

```
x=min(perim_array(1:max(max(dd)))):(max(perim_array(1:max(max(dd))))-
   min(perim_array(1:max(max(dd))))/20:max(perim_array(1:max(max(dd))));
%20 is number of bins
hist(perim_array(1:max(max(dd))),x);
title('Histogram of Circular Hole Perimeters'); %perimeter histogram
xlabel('Length of Hole Perimeters (# of Pixels)');
ylabel('Frequency');

% HOLE CIRCULARITY
for v=1:max(max(ff))
  circularity= ((perim_array(v))*perim_array(v))/area_array(v);
  circularity_array(v)=circularity;
end
x=4*pi:(max(circularity_array(1:max(max(dd))))-4*pi)/
      20:max(circularity_array(1:max(max(dd))));
%7 is number of bins
hist(circularity_array(1:max(max(dd))),x);
title('Histogram of Circular Hole Circularity'); %circularity histogram
xlabel('Circularity of Holes');
ylabel('Frequency');

%DIAMETER

diam=0;
for i=1:max(max(dd))
[r,c]=size(dd);
holerow=0;
holecol=0;
initrow=0;
initcol=0;
for j=1:r
  for k=1:c
    if dd(j,k)==i
       diam=diam+1;
       holerow=1;
    end
  end
  if (holerow==1)&(initrow==0)
    initrow=j;
  end
  holediam_array(j)=diam;
  diam=0;
end
for k=1:c
    for j=1:r
```

```
      if dd(j,k)==i
   holecol=1;
      end
    end
    if (holecol==1)&(initcol==0)
       initcol=k;
    end
  end
hole_diam=max(holediam_array);
diam_array(i)=hole_diam;
center_matrix(i,1)=(round(initrow+(hole_diam/2)));
  center_matrix(i,2)=(round(initcol+(hole_diam/2)));
end

x=min(diam_array(1:max(max(dd)))):(max(diam_array(1:max(max(dd))))-
   min(diam_array(1:max(max(dd))))/20:max(diam_array(1:max(max(dd))));
%20 is number of bins
hist(diam_array(1:max(max(dd))),x);
title('Histogram of Circular Hole Diameters'); %diameter histogram
xlabel('Length of Hole Diameters (# of Pixels)');
ylabel('Frequency');

%VORONOI CELLS

x=center_matrix(:,1);
y=center_matrix(:,2);
voronoi(x,y);
title('Circular Voronoi Cells');
D=delaunay(x,y);
for k=1:max(max(dd))
  s(k)=sum(sum(D==k));
end
mean(s(1:max(max(dd))));
hist(s(1:max(max(dd))));
title('Histogram of Edges of Circular Voronoi Cells');

%HOLE DENSITY---8x8 Grid
[r,c]=size(dd);
x=1;
y=1;
center_count=0;
while (x<=(r-49))
  y=1;
  while (y<=(c-49))
     center_count=0;
    for i=1:num_holes
```

```
      if ((center_matrix(i,1)>=x)&(center_matrix(i,1)<(x+50)))
 if ((center_matrix(i,2)>=y)&(center_matrix(i,2)<(y+50)))
   center_count=center_count+1;
 end
      end
    end

    center_countmatrix((round(x/50)+1),(round(y/50)+1))=center_count/(50*50);
    y=y+50;

  end
  x=x+50;
end

x=min(min(center_countmatrix)):(max(max(center_countmatrix))-
   min(min(center_countmatrix)))/20:max(max(center_countmatrix));
%7 is number of bins
hist(center_countmatrix(1:8,1:8),x);
title('Histogram of Circular Density-8x8 Grid'); %diameter histogram
xlabel('Density of Block');
ylabel('Frequency');

%HOLE DENSITY---36x36 Grid
tot=0;
[r,c]=size(dd);
x=1;
y=1;
center_count=0;
while (x<=(r-49))
  y=1;
  while (y<=(c-49))
     center_count=0;
    for i=1:size(center_matrix)
      if ((center_matrix(i,1)>=x)&(center_matrix(i,1)<(x+50)))
 if ((center_matrix(i,2)>=y)&(center_matrix(i,2)<(y+50)))
   center_count=center_count+1;
 end
      end
    end

    center_countmatrix((round(x/10)+1),(round(y/10)+1))=center_count;
    tot=tot+(center_count);
    y=y+10;

  end
  x=x+10;
```

```
end

POISSON DISTRIBUTION
avg=tot/1296;
r=poissrnd(avg*ones(1,1296));
hist(r);
title('Poisson Distribution with Lambda=1.1304 (Circular)');
xlabel('Number of Centers Per Block');
ylabel('Frequency');


x=min(min(center_countmatrix)):(max(max(center_countmatrix))-
 min(min(center_countmatrix)))/20:max(max(center_countmatrix));
%20 is number of bins
hist(center_countmatrix,x);
title('Histogram of Circular Density-36x36 Grid'); %density histogram
xlabel('Density of Block');
ylabel('Frequency');

PLOT
h=hist(center_countmatrix,x);
hh=sum(h');
plot(hh);
title('Plot of Circular Density');
```